1. Napisać program, który dla podanego nazwiska i imienia prowadzącego zwróci liczbę przedmiotów przez niego prowadzonych. Wykorzystaj zapytanie w PL/SQL. 2.

```
set SERVEROUTPUT on
declare
w_imie varchar(256) := '&imie';
w_nazwisko varchar(256) := '&nazwisko';
wynik number(5);
begin
select count(przedmiot.nazwa) into wynik
from wykladowca
inner join zajecia on wykladowca.id_wykladowca = zajecia.id_wykladowca
inner join przedmiot on przedmiot.id_przedmiot = zajecia.id_przedmiot
where wykladowca.imie = w_imie and wykladowca.nazwisko = w_nazwisko;
DBMS_OUTPUT.PUT_LINE(wynik);
end;
```

2. Napisać program PL/SQL, który dla podanego nr albumu studenta oraz budynku i sali zwróci liczbę przedmiotów, na które uczęszcza dany student. Wykorzystaj zapytanie w PL/SQL

```
set SERVEROUTPUT on

declare
w_nrAlbumu number(10) := '&idAlbumu';
w_sala VARCHAR(256) := '&sala';
w_budynek VARCHAR(256) := '&budynek';
wynik number(10);
begin
select count(przedmiot.id_przedmiot) into wynik
from student
inner join ocena on ocena.id_student = student.id_student
inner join zajecia on zajecia.id_zajecia = ocena.id_zajecia
inner join przedmiot on przedmiot.id_przedmiot = zajecia.id_przedmiot
inner join sala on sala.id_sala = zajecia.id_sala
inner join budynek on budynek.id_budynek = sala.id_budynek
where student.nralbumu = s_nr_albumu and budynek.id_budynek = b_id_budynku and sala.id_sala
= s_id_sali;
DBMS_OUTPUT.PUT_LINE('Wynik: ' || wynik);
end;
```

3. Napisz program, który wyświetli imiona, nazwiska oraz numery albumu wszystkich studentów, w następującej postaci (wynik powinien być posortowany według nazwisk w kolejności odwrotnej, wszystkie nazwiska dużymi literami, odstępy w postaci wykropkowanej). Wykorzystaj polecenie FETCH.

```
set SERVEROUTPUT on

declare
w_idStudent student.id_student%type;
w_imieStudent student.imie%TYPE;
w_nazwiskoStudent student.nazwisko%type;
w_adres student.id_adres%type;
w_nrAlbumu student.nralbumu%type;
w_idGrupa student.id_grupa%type;

Cursor grupa is
select imie,upper(nazwisko),nralbumu
from student
order by nazwisko desc;

begin
open grupa;
loop
fetch grupa into w_imieStudent,w_nazwiskoStudent,w_nrAlbumu;
DBMS_OUTPUT.PUT_LINE(Rpad(w_imieStudent,15,'.') ||  Rpad(w_nazwiskoStudent,15,'.') ||
w_nralbumu);
exit when grupa%notfound;
end loop;
close grupa;
end;
```

4. Napisz program w trzech wariantach, który pobierze dane (nazwisko, imię, numer albumu) wszystkich studentów i umieści je w zmiennej rekordowej. W pierwszym wariancie zadeklaruj zmienną jako rekord typu TYPE … IS RECORD, w drugim wykorzystaj atrybut %ROWTYPE, w trzecim wykorzystaj pętlę FOR z podzapytaniem. Wyświetl wszystkie dane z rekordu.

a)

```
set SERVEROUTPUT on

declare
type studenci is RECORD(
w_imie student.imie%type,
w_nazwisko student.nazwisko%type,
w_nrAlbumu student.nralbumu%type
);

cursor cStudenci is
select imie,nazwisko,nralbumu from student;
vart studenci;
begin
open cStudenci;
loop
fetch cStudenci into vart;
exit when cStudenci%notfound;
DBMS_OUTPUT.PUT_LINE(vart.w_imie || ' ' || vart.w_nazwisko || ' ' || vart.w_nrAlbumu);
end loop;
end;
```

b)

```
set SERVEROUTPUT on

declare
cursor cursorStudent is
select imie,nazwisko,nralbumu from student;
wynik cursorStudent%rowtype;

begin
open cursorStudent;
loop
fetch cursorStudent into wynik;
exit when cursorStudent%notfound;
DBMS_OUTPUT.PUT_LINE(wynik.imie || ' ' || wynik.nazwisko || ' ' ||  wynik.nralbumu);
end loop;
end;
```

c)

```
set SERVEROUTPUT on

declare
begin
for licz in(
select imie,nazwisko,nralbumu from student)
loop
dbms_output.put_line(licz.imie || ' ' || licz.nazwisko || ' ' || licz.nralbumu);
end loop;
end;
```

5.

Napisz program, który wyświetli dane o wszystkich studentach, którzy nie mają zaliczenia przynajmniej z jednego przedmiotu (nazwa przedmiotu, nazwisko i imię prowadzącego). Korzystając z atrybutu %ROWCOUNT ogranicz ilość wyników do trzech pozycji.

```
set SERVEROUTPUT on
declare
cursor studentCursor is
select przedmiot.nazwa,student.imie, student.nazwisko from student
INNER join ocena on ocena.id_student = student.id_student
inner join zajecia on zajecia.id_zajecia = ocena.id_zajecia
inner join przedmiot on przedmiot.id_przedmiot = zajecia.id_przedmiot
where ocena.ocena = 2;

begin
for i in studentCursor
loop
exit when studentCursor%rowcount = 4;
DBMS_OUTPUT.PUT_LINE(i.nazwa || ' '  || i.imie || ' '  || i.nazwisko);
end loop;
end;
```

6.
Podnieś ocenę wszystkim studentom z przedmiotu (nazwa przedmiotu – parametrem kursora) o jeden stopień, a pozostałe oceny o 0,5 jeśli to możliwe. Bezpośrednio po zmianie oceny wyświetl na ekranie nazwisko i imię studenta oraz nową ocenę. W pierwszym wariancie programu skorzystaj z klauzuli WHERE CURRENT OF (celem wskazania rekordu do zmiany ceny). W wariancie drugim skorzystaj z klauzuli RETURNING INTO (celem przekazania zmienionej oceny do zmiennej).

```
set SERVEROUT on
declare
przedmiotWartosc przedmiot.nazwa%type := '&przedmiotNazwa';

cursor studentCursor(przedmiotU przedmiot.nazwa%type) is
select przedmiot.nazwa,ocena.ocena, student.imie, student.nazwisko from student
inner join ocena on ocena.id_student = student.id_student
inner join zajecia on zajecia.id_zajecia = ocena.id_zajecia
inner join przedmiot on przedmiot.id_przedmiot = zajecia.id_przedmiot
where przedmiot.nazwa like przedmiotWartosc
for update;

begin
for i in studentCursor(przedmiotWartosc)
loop
if i.ocena <= 4 and i.nazwa = przedmiotWartosc then
update ocena set ocena = ocena + 1 where current of studentCursor;
DBMS_OUTPUT.PUT_LINE(i.nazwisko || ' ' || i.imie || ' ' || i.ocena);
elsif i.ocena <= 4.5 then
update ocena set ocena = ocena + 0.5 where current of studentCursor;
DBMS_OUTPUT.PUT_LINE(i.nazwisko || ' ' || i.imie || ' ' || i.ocena);
end if;
end loop;
end;
```

7. Napisz program, który wyświetli dane o wszystkich studentach, którzy nie mają zaliczenia przynajmniej z jednego przedmiotu (nazwa przedmiotu, nazwisko i imię prowadzącego). Korzystając z atrybutu %ROWCOUNT ogranicz ilość wyników do trzech pozycji. Uwaga – brak zaliczenia to brak wpisanej oceny z danego przedmiotu.

```
set SERVEROUT on
declare
CURSOR studentCursor is
select wykladowca.imie,wykladowca.nazwisko, przedmiot.nazwa
from przedmiot
inner join zajecia on zajecia.id_przedmiot = przedmiot.id_przedmiot
inner join wykladowca on wykladowca.id_wykladowca = zajecia.id_wykladowca
inner join ocena on ocena.id_zajecia = zajecia.id_zajecia
where ocena.ocena is NULL;
begin
for i in studentCursor
loop
dbms_output.PUT_LINE(i.imie || ' ' || i.nazwisko || ' ' || i.nazwa);
exit when studentCursor%rowcount = 4 or studentCursor%notfound;
end loop;
end;
```

8.

Podnieś ocenę wszystkim studentom z przedmiotu (nazwa przedmiotu – parametrem kursora) o jeden stopień, a pozostałe oceny o 0,5 jeśli to możliwe. Bezpośrednio po zmianie oceny wyświetl na ekranie nazwisko i imię studenta oraz nową ocenę. W pierwszym wariancie programu skorzystaj z klauzuli WHERE CURRENT OF (celem wskazania rekordu do zmiany ceny). W wariancie drugim skorzystaj z klauzuli RETURNING INTO (celem przekazania zmienionej oceny do zmiennej).

a)

```
set SERVEROUT on
declare
przedmiotNazwa przedmiot.nazwa%type := '&przedmiotNazwa';
cursor studentCursor(parNazwa przedmiot.nazwa%type) is
select ocena.ocena, przedmiot.nazwa
from ocena
inner join zajecia on ocena.id_zajecia = zajecia.id_zajecia
inner join przedmiot on przedmiot.id_przedmiot = zajecia.id_przedmiot
where przedmiot.nazwa = parNazwa
for update of ocena;
begin
for i in studentCursor(przedmiotNazwa)
loop
if i.ocena <= 4 then
update ocena set ocena.ocena = ocena.ocena + 1 where current of studentCursor;
DBMS_OUTPUT.PUT_LINE('Nowa ocena: ' || i.OCENA);
ELSIF i.ocena <= 4.5 then
update ocena set ocena.ocena = ocena.ocena + 0.5 where current of studentCursor;
DBMS_OUTPUT.PUT_LINE('Nowa ocena: ' || i.OCENA);
end if;
end loop;
end;
```

b)

```
set SERVEROUT on
declare
przedmiotNazwa przedmiot.nazwa%type := '&przedmiotNazwa';
cursor studentCursor(parNazwa przedmiot.nazwa%type) is
select ocena.ocena, przedmiot.nazwa
from ocena
inner join zajecia on ocena.id_zajecia = zajecia.id_zajecia
inner join przedmiot on przedmiot.id_przedmiot = zajecia.id_przedmiot
where przedmiot.nazwa = parNazwa
for update of ocena;
begin
for i in studentCursor(przedmiotNazwa)
      loop
         if i.ocena <= 4 then
            update ocena
            set ocena.ocena = ocena.ocena
            where ocena.OCENA = ocena.ocena + 1
```

```
            returning ocena.ocena into i.ocena;
            DBMS_OUTPUT.PUT_LINE('Nowa ocena: ' || i.ocena);
        ELSIF i.ocena <= 4.5 then
            update ocena
            set ocena.ocena = ocena.ocena
            where ocena.OCENA = ocena.ocena - 0.5
            returning ocena.ocena into i.ocena;
            DBMS_OUTPUT.PUT_LINE('Nowa ocena: ' || i.ocena);
        end if;
    end loop;
end;
```

9.

Napisz program z użyciem kursora, który odczyta informacje o wszystkich profesorach i liczbie prowadzonych przez nich przedmiotów. Wprowadź obsługę błędów jeśli profesor nie prowadzi żadnego przedmiotu lub prowadzi więcej niż dwa przedmioty. Wykorzystaj pętlę FOR z kursorem a w drugim przypadku pętlę FOR z podzapytaniem.

```
set SERVEROUT on
declare
--errorr EXCEPTION;
cursor wykladowcaCursor is
select distinct wykladowca.imie,wykladowca.nazwisko, count(zajecia.id_przedmiot) ilosc from
wykladowca
inner join zajecia on zajecia.id_wykladowca = wykladowca.id_wykladowca
inner join przedmiot on przedmiot.id_przedmiot = zajecia.id_przedmiot
group by wykladowca.imie,wykladowca.nazwisko;
begin
for i in wykladowcaCursor
loop
if i.ilosc = 1 or i.ilosc = 2 then
DBMS_OUTPUT.PUT_LINE(i.imie || ' ' || i.nazwisko || ': ' || i.ilosc);
ELSIF i.ilosc = 0 or i.ilosc > 2 then
--RAISE errorr;
DBMS_OUTPUT.PUT_LINE(i.imie || ' ' || i.nazwisko || ': ' || 'zla ilosc przedmiotow');
end if;
end loop;
--EXCEPTION
--when errorr then
--DBMS_OUTPUT.PUT_LINE('errort');
end;
```

10. Napisz blok PL/SQL służący do wprowadzania nowego studenta. Najpierw sprawdź czy dane są poprawne korzystając także z obsługi wyjątków. Po wykonaniu instrukcji INSERT spytaj użytkownika, czy chce zatwierdzić zmiany. Jeśli odpowiedź będzie NIE, wycofaj zmianę; jeśli odpowiedź będzie TAK, zatwierdź zmianę i wypisz wprowadzone wartości na ekran.

```
SET SERVEROUTPUT ON;
DECLARE
 idWartosc student.id_student%type := &id;
 imieWartosc student.imie%type := '&imieWartosc';
 nazwiskoWartosc student.nazwisko%type := '&nazwiskoWartosc';
 id_adresWartosc student.id_adres%type := '&id_adresWartosc';
 nr_albumuWartosc student.nralbumu%type := '&nr_albumuWartosc';
 id_grupyWartosc student.id_grupa%type := '&id_grupyWartosc';
 decyzjaWartosc varchar2(64);
 ex exception;
BEGIN
 insert into STUDENT values (idWartosc, imieWartosc, nazwiskoWartosc, id_adresWartosc,
nr_albumuWartosc, id_grpyWartosc);
 decyzjaWartosc := '&tak/nie';
 IF decyzjaWartosc = 'tak' then commit;
 elsif decyzjaWartosc = 'nie' then rollback;
 else raise ex;
 END IF;
EXCEPTION
 WHEN ex THEN DBMS_OUTPUT.PUT_LINE('Invalid decision input');
END;
```

11 Napisz anonimowy blok PL/SQL i zdefiniuj kursor z parametrami. Jego zadaniem ma być wyświetlenie wszystkich ocen wystawionych w podanym okresie (np. zmienne DATA_OD, DATA_DO). Wyświetlając dane o ocenach pobieraj również dane o studencie, którego dotyczą oceny oraz dane o pracowniku wystawiającym oceny

```
SET SERVEROUTPUT ON;
DECLARE
CURSOR oceny(od VARCHAR2, do VARCHAR2) IS
 SELECT student.imie s_imie, student.nazwisko s_nazwisko, wykladowca.imie w_imie,
 wykladowca.nazwisko w_nazwisko, ocena
 FROM ocena
 JOIN student USING(id_student)
 JOIN zajecia USING(id_zajecia)
 JOIN wykladowca USING(id_wykladowca)
 WHERE DATA>=TO_DATE(od,'DD/MM/YY') and DATA<=TO_DATE(do,'DD/MM/YY');
BEGIN
 FOR licz IN oceny('&od','&do')
 LOOP
 dbms_output.put_line('Student: '||' '||licz.s_imie||' '||licz.s_nazwisko||', wykladowca: '||licz.w_imie||' '||
licz.w_nazwisko||', ocena: '||licz.ocena);
 END LOOP;
END;
```

12. Napisz procedurę sparametryzowaną, w której zostanie wybrany najlepszy student, a jego nazwisko, imię i numer albumu oraz średnia zostaną przekazane do środowiska wywołującego, gdzie należy wypisać je na ekranie. Wprowadź obsługę błędów, jeśli więcej niż jeden student uzyska najwyższą średnią.

```
create or replace PROCEDURE najlepszy (ocenaWartosc in number) is
ilosc number :=0;
cursor studentCursor is
select imie,nazwisko,nralbumu,avg(ocena) as srednia
from student
inner join ocena on ocena.id_student = student.id_student
where ocena.ocena = ocenaWartosc
group by imie,nazwisko,nralbumu order BY srednia desc;
begin
for i in studentCursor
loop
ilosc := ilosc +1;
end loop;
for i in studentCursor
loop
if ilosc = 0 then
DBMS_OUTPUT.PUT_LINE(i.imie || ' ' || i.nazwisko || ' ' || i.nralbumu || ' ' || i.srednia);
else
DBMS_OUTPUT.PUT_LINE('Wiecej niz 1 student');
EXIT;
end if;
end loop;
end;

EXECUTE najlepszy(5);
```

13. Napisz funkcję, która dla podanego nazwiska i imienia prowadzącego (parametr), zwróci liczbę przedmiotów przez niego prowadzonych

```
create or replace function LICZBAPRZEDMIOTOW(
imieWartosc in varchar2,
nazwiskoWartosc in varchar2
) return VARCHAR2 is
wynik number(5);
begin
select count(zajecia.id_przedmiot)
into wynik
from wykladowca
inner join zajecia on
zajecia.id_wykladowca = wykladowca.id_wykladowca
inner join przedmiot on
przedmiot.id_przedmiot = zajecia.id_przedmiot
where imie like imieWartosc and nazwisko like nazwiskoWartosc;
RETURN wynik;
end LICZBAPRZEDMIOTOW;

select distinct LICZBAPRZEDMIOTOW('Agata','Krog') from wykladowca;
```

14 Napisać funkcje PL/SQL, która dla podanego nr albumu studenta oraz budynku i sali (parametry) zwróci liczbę przedmiotów, na które uczęszcza dany student.

```
create or replace FUNCTION LICZBAPRZEDMIOTOW(
albumWartosc in number,
budynekWartosc in number,
salaWartosc in number
) return number is
wynik number(6);
begin
select count(zajecia.id_przedmiot) into wynik from student
inner join ocena on
ocena.id_student =  student.id_student
inner join zajecia on
zajecia.id_zajecia = ocena.id_zajecia
inner join sala on
sala.id_sala = zajecia.id_sala
inner join budynek on
budynek.id_budynek = sala.id_budynek
where
albumWartosc = nralbumu and
budynekWartosc = budynek.id_budynek and
salaWartosc = sala.id_sala;
return wynik;
end LICZBAPRZEDMIOTOW;

select distinct LICZBAPRZEDMIOTOW(92838,122,124) from student;
```

15 Napisać procedurę, która zmodyfikuje typ zajęć dla podanej, jako parametr grupy studenckiej w zależności od jego aktualnej zawartości. Wykorzystaj klauzurę: L – W oraz W – Ć

```
create or replace procedure ZMIEN(
grupaWartosc in varchar2
) is
idGrupyWartosc number;
begin
select grupa.id_grupa into idGrupyWartosc
from grupa where grupa.nazwa = grupaWartosc;
update zajecia
set id_charakter = 3
where id_charakter = 4 and id_charakter = 1 and id_grupa = idGrupyWartosc;
end ZMIEN;



--wyklad 3
--labolatoria 1
--lektorat 4
--L -> W 4 = 3 and 1=3
--W -> Ć 3 = 6


begin
ZMIEN('GRUPA 10n1');
end;
```

16. Napisać procedurę PL/SQL, która dla podanego budynku, sali (parametry), wypisze wszystkie prowadzone zajęcia - tytuł, nazwisko i imię prowadzącego, nazwę przedmiotu, typ zajęć, nazwę grupy studenckiej.

```
set SERVEROUT on
create or replace procedure wyswietl(
budynekWartosc in varchar2,
salaWartosc in varchar2
) is
cursor studentCursor is
select imie as imieWykladowca,
nazwisko as nazwiskoWykladowca,
tytulnaukowy.nazwa as nazwaTytulnaukowy,
przedmiot.nazwa as nazwaPrzedmiot,
charakter.nazwa as nazwaCharakter,
grupa.nazwa as nazwaGrupa
from zajecia
inner join grupa on
grupa.id_grupa = zajecia.id_grupa
inner join charakter on
charakter.id_charakter = zajecia.id_charakter
inner join wykladowca on
wykladowca.id_wykladowca = zajecia.id_wykladowca
inner join tytulnaukowy on
tytulnaukowy.id_tytul = wykladowca.id_tytul
inner join przedmiot on
przedmiot.id_przedmiot = zajecia.id_przedmiot
inner join sala on
sala.id_sala = zajecia.id_sala
inner join budynek on
budynek.id_budynek = sala.id_budynek
where budynek.nazwa = budynekWartosc and sala.kodsali = salaWartosc;
ilosc number := 0;
begin
for i in studentCursor
loop
ilosc := ilosc + 1;
end loop;
if ilosc > 0 then
for i in studentCursor
loop
DBMS_OUTPUT.PUT_LINE(i.nazwaTytulnaukowy || ' ' || i.imieWykladowca || ' ' ||
i.nazwiskoWykladowca || ' ' || i.nazwaPrzedmiot
|| ' ' || i.nazwaCharakter || ' ' || i.nazwaGrupa );
end loop;
ELSE
DBMS_OUTPUT.PUT_LINE('BRAK');
end if;
end wyswietl;


EXECUTE wyswietl('Wydzia3 Informatyki i Telekomunikacji','F201');
```

17. Napisz procedurę, która dla podanego wykładowcy (imię i nazwisko – parametry procedury) wyświetli imiona, nazwiska oraz numery albumu wszystkich studentów, którzy mieli zajęcia z tym wykładowcą w sali (parametr). Dane mają być posortowane alfabetycznie według nazwisk studentów. Utwórz tabelę HISTORIA i dodaj do niej zestaw rekordów zwrócony przez zapytanie.

```
create or replace procedure pokaz(
imieWartosc in varchar2,
nazwiskoWartosc in VARCHAR2,
salaWartosc in varchar2
) is
cursor studentCursor is
select student.imie as imieStudent,student.nazwisko as nazwiskoStudent,student.nralbumu as
nralbumuStudent
from student
inner join ocena on
ocena.id_student = student.id_student
inner join zajecia on
zajecia.id_zajecia = ocena.id_zajecia
inner join wykladowca on
wykladowca.id_wykladowca = zajecia.id_wykladowca
inner join sala on
sala.id_sala = zajecia.id_sala
where
wykladowca.imie = imieWartosc and
wykladowca.nazwisko = nazwiskoWartosc and
sala.kodsali = salaWartosc
order by nazwiskoStudent;
begin
for i in studentCursor
loop
DBMS_OUTPUT.PUT_LINE(i.imieStudent || ' ' || i.nazwiskoStudent  || ' ' || i.nralbumuStudent);
insert into historia (imie,nazwisko,nr_albumu) values
(i.imieStudent,i.nazwiskoStudent,i.nralbumuStudent);
end loop;
end pokaz;

EXECUTE pokaz('Piotr','Breit','Hala');
```

18. Napisz procedurę w dwóch wariantach, która przeliczy stopnie Fahrenheita na Celsjusza według wzoru [ºC]=([ºF]-32)*5/9. W pierwszym wariancie procedura niech przyjmuje dwa parametry (Fahrenheit i Celsjusz), w drugim – jeden (temperatura). Skorzystaj z parametrów IN, OUT, IN OUT. Przetestuj jej działanie w bloku anonimowym.

```
--wersja 1

set SERVEROUT on
create or replace PROCEDURE temperatura(FahrenheitWartosc in number,
CelsjuszWartosc out number
)is
begin
CelsjuszWartosc := (FahrenheitWartosc-32)*(5/9);
end;


declare
fahrenheit number := 25;
celsjusz number;
begin
temperatura(fahrenheit,celsjusz);
DBMS_OUTPUT.PUT_LINE(celsjusz);
end;

--wersja 2

set SERVEROUT on
create or replace PROCEDURE temperatura2(FahrenheitWartosc in out number
)is
begin
FahrenheitWartosc := (FahrenheitWartosc-32)*(5/9);
end;

declare
fahrenheit number := 25;
begin
temperatura2(fahrenheit);
DBMS_OUTPUT.PUT_LINE(fahrenheit);
end;
```

19. Utwórz sekwencję, która pozwoli na nadawanie kolejnych numerów dla wstawianych ocen oraz studentów. Dodaj po dwa rekordy do tabeli student oraz oceny .

```
CREATE SEQUENCE student_seq
START WITH 52
INCREMENT BY 1;

INSERT INTO STUDENT VALUES (student_seq.NEXTVAL, 'Andrzej', 'Grabowski', 1, 22222, 5);
INSERT INTO STUDENT VALUES (student_seq.NEXTVAL, 'Michał', 'Szpak', 2, 33333, 6);

CREATE SEQUENCE ocena_seq
START WITH 36
INCREMENT BY 1;

INSERT INTO OCENA VALUES (ocena_seq.NEXTVAL, 52, 3, 3.5, '20/03/10');
INSERT INTO OCENA VALUES (ocena_seq.NEXTVAL, 53, 5, 4, '18/12/16');
```

20. Utwórz wyzwalacz, który przy próbie zmiany oceny sprawdzi, czy nowa ocena nie jest zwiększona więcej niż 0.5. Jeśli tak wypisze starą i nową ocenę. Dodaj też odpowiedni komunikat.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER zmianaOceny
BEFORE UPDATE OF ocena ON Ocena FOR EACH ROW
DECLARE
zmiana NUMBER(2,1);
BEGIN
zmiana := :NEW.ocena - :OLD.ocena;
IF(zmiana > 0.5) THEN
   DBMS_OUTPUT.PUT_LINE('Nowa ocena: ' || :NEW.ocena || ' stara ocena: ' || :OLD.ocena);
END IF;
END;

UPDATE Ocena SET ocena = 3.5 where id_ocena = 4;
```

21. Napisz funkcję SILNIA w sposób rekurencyjny. Przetestuj jej działanie

```
create or replace FUNCTION silnia(x number)
return number is
f number;
begin
if x = 0 then
   f := 1;
else
   f := x * silnia(x-1);
end if;
return f;
end;

declare
liczba number;
wynik number;
begin
liczba := '&podajLiczbe';
wynik := silnia(liczba);
DBMS_OUTPUT.PUT_LINE('Silnia z ' || liczba || ' wynosi ' || wynik);
end;
```

22. Utworzyć pakiet o nazwie MOJ_PAKIET oraz zaimplementować podane niżej procedury i funkcje ▪ DODAJ_STUD zadaniem procedury ma być dodawanie nowego studenta do tabeli student. Numer ID powinien być pobierany automatycznie ze zdefiniowanej w tym celu sekwencji. ▪ ZMIEN_STUD – procedura modyfikuje dane wskazanego studenta. ▪ USUN_STUD – procedura kasuje dane wskazanego studenta. ▪ ZMIEN_ADRES – zadaniem tej procedury jest zmiana adresu wskazanego studenta. ▪ TOP_N – wyświetla listę N studentów (N podawane jako parametr wejściowy procedury), którzy mają najwyższe średnie. Dane o tych studentach (imię, nazwisko, adres, średnia ocen) powinni zostać dodatkowo zapisani do tabeli o nazwie TOP_N_STUD (tabelę tą trzeba utworzyć bezpośrednio w kodzie procedury). ▪ ZMIANA – procedura zmienia miejsce i czas prowadzenia przedmiotu przez konkretnego wykładowcę) nazwa przedmiotu, imię, nazwisko wykładowcy to parametry procedury. ▪ WYPISZ– zadaniem procedury jest wypisanie średniej oceny dla wybranego przedmiotu, którego nazwa będzie parametrem.. ▪ MINMAX – zadaniem funkcji jest zwrócenie wartości (w zależności od podanego parametru) maksymalnej, minimalnej, średniej dla danej grupy (parametr). Funkcja powinna przyjmować tylko jeden z dwóch parametrów: MAX, MIN, Podanie innego parametru powinno wygenerować stosowne ostrzeżenie.

```
CREATE OR REPLACE PACKAGE MOJ_PAKIET IS
PROCEDURE DODAJ_STUD(imie IN VARCHAR, nazwisko IN VARCHAR, id_adres IN
NUMBER, album IN NUMBER, id_grupa IN NUMBER);
PROCEDURE ZMIEN_STUD(id_student IN NUMBER, nazwisko IN VARCHAR);
PROCEDURE USUN_STUD(id_student IN NUMBER);
PROCEDURE ZMIEN_ADRES(id_student IN NUMBER, ulica IN VARCHAR,
   nr_bud in NUMBER, nr_lok IN NUMBER, kod_poczt IN VARCHAR, miasto IN VARCHAR);
PROCEDURE TOP_N(N IN NUMBER);
PROCEDURE ZMIANA(nazwa_p IN VARCHAR, imie IN VARCHAR, nazwisko IN VARCHAR);
PROCEDURE WYPISZ(nazwa_przedmiotu IN VARCHAR);
FUNCTION MINMAX(p IN VARCHAR, grupa IN VARCHAR) RETURN NUMBER;
END;

CREATE OR REPLACE PACKAGE BODY MOJ_PAKIET IS
PROCEDURE DODAJ_STUD(imie IN VARCHAR, nazwisko IN VARCHAR, id_adres IN
NUMBER, album IN NUMBER, id_grupa IN NUMBER) IS
BEGIN
   INSERT INTO STUDENT VALUES (student_seq.NEXTVAL, imie, nazwisko, id_adres, album,
id_grupa);
END DODAJ_STUD;
PROCEDURE ZMIEN_STUD(id_student IN NUMBER, nazwisko IN VARCHAR) IS
nazwisko_h VARCHAR(30) := nazwisko;
id_h NUMBER(10, 0) := id_student;
BEGIN
   UPDATE Student SET nazwisko = nazwisko_h where id_student = id_h;
END ZMIEN_STUD;
PROCEDURE USUN_STUD(id_student IN NUMBER) IS
id_h NUMBER(10, 0) := id_student;
BEGIN
   DELETE FROM Student WHERE id_student = id_h;
END USUN_STUD;
PROCEDURE ZMIEN_ADRES(id_student IN NUMBER, ulica IN VARCHAR, nr_bud in
NUMBER,
```

```
   nr_lok IN NUMBER, kod_poczt IN VARCHAR, miasto IN VARCHAR) IS
ulica_h VARCHAR(30) := ulica;
id_stud_h NUMBER(10, 0) := id_student;
nr_bud_h NUMBER(4, 0) := nr_bud;
nr_lok_h NUMBER(4, 0) := nr_lok;
kod_poczt_h VARCHAR2(6) := kod_poczt;
miasto_h VARCHAR(50) := miasto;
BEGIN
   UPDATE Adres SET ulica = ulica_h, nrbudynku = nr_bud_h, nrlokalu = nr_lok_h,
      kodpocztowy = kod_poczt_h, miasto = miasto_h WHERE id_adres = (
         SELECT a.id_adres FROM Student s, Adres a
         WHERE s.id_student = id_stud_h AND a.id_adres = s.id_adres
      );
END ZMIEN_ADRES;
PROCEDURE TOP_N(N IN NUMBER) IS
N_h NUMBER(5, 0) := N;
tabela_srednia VARCHAR2(4000);
id_top1 NUMBER(4, 0) := 0;
CURSOR top_srednia_cursor IS
   SELECT AVG(o.ocena) as srednia, s.imie, s.nazwisko, a.id_adres FROM Ocena o, Student s,
Adres a
   WHERE o.id_student = s.id_student AND a.id_adres = s.id_adres
   GROUP BY s.imie, s.nazwisko, a.id_adres
   ORDER BY AVG(o.ocena) DESC;
BEGIN
   EXECUTE IMMEDIATE 'CREATE TABLE TOP_N_STUD (
                  id_top NUMBER(4, 0),
                  imie VARCHAR2(20),
                  nazwisko VARCHAR2(30),
                  id_adres NUMBER(4, 0),
                  srednia NUMBER(4,2))';
   FOR licz IN top_srednia_cursor
   LOOP
      EXECUTE IMMEDIATE 'INSERT INTO TOP_N_STUD VALUES (:id_top, :imie, :nazwisko,
:id_adres, :srednia)'
      USING id_top1, licz.imie, licz.nazwisko, licz.id_adres, licz.srednia;
      id_top1 := id_top1 + 1;
      EXIT WHEN top_srednia_cursor%ROWCOUNT = N_h;
   END LOOP;
END TOP_N;
ROCEDURE ZMIANA(nazwa_p IN VARCHAR, imie IN VARCHAR, nazwisko IN VARCHAR)
IS
CURSOR przedmiot_cursor IS
   SELECT z.id_sala, z.dzientyg, p.id_przedmiot FROM Wykladowca w, Przedmiot p, Zajecia z
   WHERE z.id_wykladowca = w.id_wykladowca AND z.id_przedmiot = p.id_przedmiot
   AND w.imie = imie AND w.nazwisko = nazwisko AND p.nazwa = nazwa_p;
BEGIN
   FOR licz in przedmiot_cursor
   LOOP
       UPDATE Zajecia SET dzientyg = 'WTO', id_sala = 19
       WHERE id_przedmiot = licz.id_przedmiot AND dzientyg = licz.dzientyg AND id_sala =
licz.id_sala;
```

```
   END LOOP;
END ZMIANA;
PROCEDURE WYPISZ(nazwa_przedmiotu IN VARCHAR) IS
przedmiot VARCHAR2(50) := nazwa_przedmiotu;
srednia NUMBER(3,2);
BEGIN
   SELECT AVG(o.ocena) INTO srednia FROM Przedmiot p, Ocena o, Zajecia z
   WHERE p.id_przedmiot = z.id_przedmiot AND o.id_zajecia = z.id_zajecia AND p.nazwa =
przedmiot
   GROUP BY p.id_przedmiot;
   DBMS_OUTPUT.PUT_LINE('Srednia z ' || przedmiot || ' to ' || srednia);
END WYPISZ;
FUNCTION MINMAX(p IN VARCHAR, grupa IN VARCHAR) RETURN NUMBER IS
wynik NUMBER(4, 2) := 0;
BEGIN
   IF p = 'MIN' THEN
      SELECT MIN(o.ocena) INTO wynik FROM Ocena o, Student s, Grupa g
      WHERE o.id_student = s.id_student AND g.id_grupa = s.id_grupa AND g.nazwa = grupa
      GROUP BY g.nazwa;
   ELSIF p = 'MAX' THEN
      SELECT MAX(o.ocena) INTO wynik FROM Ocena o, Student s, Grupa g
      WHERE o.id_student = s.id_student AND g.id_grupa = s.id_grupa AND g.nazwa = grupa
      GROUP BY g.nazwa;
   ELSIF p = 'AVG' THEN
      SELECT AVG(o.ocena) INTO wynik FROM Ocena o, Student s, Grupa g
      WHERE o.id_student = s.id_student AND g.id_grupa = s.id_grupa AND g.nazwa = grupa
      GROUP BY g.nazwa;
   ELSE
      DBMS_OUTPUT.PUT_LINE('Niepoprawne polecenie');
   END IF;
   RETURN wynik;
END MINMAX;
END MOJ_PAKIET;

SET SERVEROUTPUT ON;
DECLARE
BEGIN
   MOJ_PAKIET.DODAJ_STUD('Witold', 'Hrabia', 1, 444444, 5);
   MOJ_PAKIET.ZMIEN_STUD(1, 'Milan');
   MOJ_PAKIET.USUN_STUD(54);
   MOJ_PAKIET.ZMIEN_ADRES(1, 'Wesola', 5, 10, '31-323', 'Warszawa');
   MOJ_PAKIET.TOP_N(5);
   MOJ_PAKIET.ZMIANA('Wychowanie Fizyczne', 'Piotr', 'Breit');
   MOJ_PAKIET.WYPISZ('Wychowanie Fizyczne');
   DBMS_OUTPUT.PUT_LINE(MOJ_PAKIET.MINMAX('AVG', 'GRUPA 21'));
END;
```

1. Napisz procedurę wprowadzającą do tabeli WYPOŻYCZENIA rekord rejestrujący nowe wypożyczenie. Aby można było zarejestrować nowe wypożyczenie należy sprawdzić, czy czytelnik o podanym nazwisku i imieniu (parametry procedury) znajduje się już w bazie danych. Jeśli nie należy wprowadzić nowego czytelnika do tabeli CZYTELNICY. Zaimplementuj obsługę błędów. 4 pkt.

```
set SERVEROUTPUT on

create or REPLACE PROCEDURE wprowadz(w_imie in VARCHAR2,w_nazwisko in VARCHAR2,w_tytul in VARCHAR2) is
CURSOR czytelnikCursor is
select count(id_czyt),id_czyt from czytelnik
where czytelnik.imie = upper(w_imie) and czytelnik.nazwisko = upper(w_nazwisko)
group by id_czyt;
czytelnikIlosc number :=0;
idCzytelnik number := 0;
idKsiazki number := 0;
errorInfo EXCEPTION;

CURSOR cursorIdKsiazki is
select id_ks from ksiazka where tytul = upper(w_tytul);

begin
open czytelnikCursor;
fetch czytelnikCursor into czytelnikIlosc,idczytelnik;
open cursorIdKsiazki;
fetch cursorIdKsiazki into idksiazki;
if czytelnikilosc = 0 then
insert into czytelnik (id_czyt,nazwisko,imie,kod_pocztowy,miejscowosc,ulica,telefon)
values ((select max(id_czyt)+1 from czytelnik),w_nazwisko,w_imie,'38-400','Kraków','polna','123123123');
insert into wypozyczenia (ID_WYP,ID_KS,ID_CZYT,DATA_WYP,DATA_ZWR) values
((select max(id_wyp)+1 from wypozyczenia),idksiazki,(select max(id_czyt) from czytelnik),TO_CHAR(sysdate, 'YY/MM/DD'),TO_CHAR(sysdate+30, 'YY/MM/DD'));
elsif czytelnikilosc = 1 then
insert into wypozyczenia (ID_WYP,ID_KS,ID_CZYT,DATA_WYP,DATA_ZWR) values
((select max(id_wyp)+1 from wypozyczenia),idksiazki,idczytelnik,TO_CHAR(sysdate, 'YY/MM/DD'),TO_CHAR(sysdate+30, 'YY/MM/DD'));
else
raise errorInfo;
end if;

EXCEPTION
when errorInfo then
DBMS_OUTPUT.PUT_LINE('Istnieje więcej takich czytelnikow');


end wprowadz;
```

```
declare
begin
wprowadz('Piotrr','kowalski','dziady');
end;
```

2. Napisz funkcję, która pobierze i wyświetli wszystkie książki, które były w stanie wypożyczenia w zadanym okresie (parametry od i do) . Książki nie mogą się powtarzać i muszą być wyświetlone w porządku odwróconym w stosunku do roku wydania. Funkcja ma zwrócić liczbę wypożyczonych książek, które zostały wydane po roku 2015.

3 pkt ‘BRAKUJE USUNIĘCIA POWTARZANIA SIĘ ’

```
set SERVEROUTPUT on

create or replace function wszystkieKsiazki(od in date,do in date)

return varchar2 is v

wynik number;

cursor cursorKsiazki is

select distinct tytul,wypozyczenia.data_wyp as dataWypozyczenia,wypozyczenia.data_zwr as dataZwrotu,ksiazka.rok_wyd

from ksiazka

inner join wypozyczenia on wypozyczenia.id_ks = ksiazka.id_ks

where wypozyczenia.data_wyp > od and wypozyczenia.data_zwr < do

order by ksiazka.rok_wyd desc;

cursor cursorIlosc is

select count(ksiazka.id_ks) from ksiazka

inner join wypozyczenia on wypozyczenia.id_ks = ksiazka.id_ks

where ksiazka.rok_wyd>'15/12/31' and (wypozyczenia.data_wyp > od and wypozyczenia.data_zwr < do);

begin

open cursorIlosc;

fetch cursorIlosc into wynik;

for i in cursorKsiazki

loop

dbms_output.put_line(i.tytul || ' ' || i.dataWypozyczenia || ' ' ||i.dataZwrotu);

end loop;

RETURN wynik;

end wszystkieKsiazki;
```

```
select distinct wszystkieKsiazki('20/01/15','20/02/16') from ksiazka;
```

3. Napisz procedurę, która dla podanego gatunku literackiego (parametr) wyświetli autora, tytuł, oraz nazwę gatunku wszystkich książek wydanych w ciągu ostatniego roku. Dane mają być posortowane alfabetycznie według nazwisk i imion autorów. Procedura jako parametr wyjściowy ma zwracać liczbę książek z podanego gatunku, która znajduje się w bibliotece. 3 pkt

```
set SERVEROUTPUT on

create or replace PROCEDURE gatunekLiteracki(gatuenkLiteracki in VARCHAR2,wynik out number)
is
CURSOR cursorAutor is
select DISTINCT
autor.imie as imieAutor,
autor.nazwisko as nazwiskoAutor,
gatunek.nazwa as nazwaGatunek,
ksiazka.tytul as tytulKsiazka
from autor
inner join autor_tytul on autor_tytul.id_autor = autor.id_aut
inner join ksiazka on ksiazka.id_ks = autor_tytul.id_ksi
inner join gatunek on gatunek.id_gat = ksiazka.id_gat
where gatunek.nazwa = gatuenkLiteracki
and ksiazka.rok_wyd >= sysdate - interval '1' year
order by nazwiskoAutor,imieAutor;

cursor iloscKsiazek is
select count(ksiazka.id_ks) from ksiazka
inner join gatunek on gatunek.id_gat = ksiazka.id_gat
where gatunek.nazwa = gatuenkLiteracki
group by gatunek.nazwa;

begin
open iloscKsiazek;
fetch iloscKsiazek into wynik;
for i in cursorAutor
loop
dbms_output.put_line(i.imieAutor || ' ' || i.nazwiskoAutor || ' ' || i.nazwaGatunek || ' ' || i.tytulKsiazka);
end loop;
end gatunekLiteracki;

declare
wynik number := 0;
begin
gatunekLiteracki('Dla dzieci',wynik);
dbms_output.put_line(wynik);
end;
```

4. Napisz procedurę wyzwalaną, która ma utrzymywać stałą liczbę wypożyczonych książek. Zmiany, które mają wpływ na liczbę wypożyczonych książek są związane z pożyczaniem i oddawaniem książki lub zakupem nowej.